In the Claims:

1. (Currently Amended) A method, comprising the steps of:

loading a code module into a memory space of a first domain, wherein the first domain owns one of a kernel space and a portion of a user space, the code module including an instruction having a symbol reference;

determining if the symbol reference is to an external location outside of the memory space;

generating a link stub for the symbol reference when the symbol reference is to an external location to access the external location;

redirecting the instruction to the link stub; and

determining if the external location is within a second domain that is within a protection view of the first domain, wherein the second domain owns the other one of the kernel space and a portion of the user space;

requesting attachment of the second domain to the first domain when the second domain is determined not to be within the protection view of the first domain; and

attaching the second domain to the first domain using an attachment mechanism.

2. (Original) The method of claim 1, wherein the link stub is part of a linking table entry corresponding to the symbol reference.

3. (Original) The method of claim 1, wherein the link stub is a jump instruction to the external location.

4. (Cancelled)

5. (Previously Presented) The method of claim 1, further comprising the steps of:

determining whether the attachment request is permitted based on authorization information provided by the first domain;

wherein attachment to the second domain is not permitted when the attachment request is not permitted.

6. (Cancelled)

7. (Cancelled)

8. (Currently Amended) A method, comprising:

creating a task in a first domain, wherein the first domain owns one of a kernel space and a portion of a user space, the task executing a number of instructions;

executing a first jump instruction in the number of instructions that refers to a link stub corresponding to an external location in a second domain, wherein the second domain owns the other one of the kernel space and a portion of the user space;

executing the link stub;

comparing the external location to a task protection view;

generating a processing exception when the external location is outside the task protection view; and

executing an exception handling routine in response to the generation of the processing exception, the exception handling routine including,

saving a pre-exception setting of the task protection view,

altering the task protection view to include a protection view of the second domain, and

jumping to the external location.

9. (Original) The method of claim 8, wherein the link stub is part of linking table entry corresponding to the external location.

10. (Previously Presented) The method of claim 8, wherein the link stub includes a second jump instruction to the external location.

11. (Cancelled)

12. (Cancelled)

13. (Previously Presented) The method of claim 8, wherein the task protection view is saved on a task protection switch stack.

14. (Previously Presented) The method of claim 8, further comprising steps of:

retrieving the pre-exception setting of the task protection view;

restoring the task protection view using the pre-exception setting of the task protection view;

returning to an a subsequent instruction to the first jump instruction in the number of instructions.

15. (Currently Amended) A computer system, comprising

a system space having a number of memory locations, wherein the system space includes a kernel space and at least one user space;

a number of protection domains, at least one of the number of protection domains owning a portion of the system space, wherein each of the number of protection domains includes a protection view defining a set of the number of protection domains to which unprotected access may be made.

16. (Original) A computer system of claim 15, wherein the at least one of the number of protection domains includes at least one of

a code module,

a link stub, and

an entry point.

17. (Original) The system of claim 16, wherein the link stub is part of a linking table entry in a linking table.

18. (Original) The system of claim 16, wherein the entry point is represented in a symbol table.

19. (Original) The system of claim 16, wherein at least one of the number of protection domains is a system protection domain that includes:

at least one code module including executable code for operating system services; and

at least one system object owned by the system protection domain.

20. (Original) The system of claim 19, wherein the system protection domain includes a protection domain list that includes entries for each of the number of protection domains.

21. (Original) The system of claim 19, wherein at least one of the number of protection domains is a first protection domain that includes:

at least one code module including executable code for a first set of functions; and

a number of link stubs;

wherein at least one of the link stubs corresponds to a symbol referenced in the executable code for the first set of functions, and such at least one link stub includes executable code to direct execution to the executable code for operating system services.

22. (Original) The system of claim 21, wherein the number of protection domains includes a second protection domain that includes:

at least one code module including executable code for a second set of functions; and

a number of entry points;

wherein each of the entry points corresponds to a symbol in the executable code for the second set of functions.

23. (Original) The system of claim 22, wherein one of the link stubs of the first protection domain corresponds to one of the number of entry points in the second protection domain, and such link stub includes executable code to direct execution to the one of the number of entry points in the second protection domain.

24. (Cancelled)

25. (Previously Presented) The system of claim 15, wherein the protection view sets a memory range of allowable memory accesses, and wherein a memory fault is generated when memory access is attempted outside of the memory range.

26. (Previously Presented) The system of claim 15, wherein the memory range is contiguous.

27. (Original) The system of claim 25, further comprising an exception handling routine that is executed on the occurrence of the memory fault, the exception handling routine including a protection switch mechanism.

28. - 34. (Cancelled)